AFRL-RI-RS-TR-2017-172

X-GRAPHS: LANGUAGE AND ALGORITHMS FOR HETEROGENEOUS GRAPH STREAMS

THE LELAND STANFORD JUNIOR UNIVERSITY

SEPTEMBER 2017

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2017-172 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
JOHN SPINA
Work Unit Manager

/ S /
MICHAEL J. WESSING
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
SEP 2017	FINAL TECHNICAL REPORT	SEP 2012 – MAR 2017

4. TITLE AND SUBTITLE

X-GRAPHS: LANGUAGE AND ALGORITHMS FOR HETEROGENEOUS GRAPH STREAMS

5a. CONTRACT NUMBER
FA8750-12-2-0335

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702E

6. AUTHOR(S)

Oyekunle Olukotun

5d. PROJECT NUMBER
XDAT

5e. TASK NUMBER
A0

5f. WORK UNIT NUMBER
13

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Leland Stanford Junior University
450 Serra Mall
Stanford, CA 94305-2004

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIEA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2017-172

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The overall goal of the X-Graphs project was to develop computational techniques and software tools for graph analytics. This report describes the two main components of this project. The first component focuses on support for interactive graph analytics applications on medium to large size graphs. The second component focuses on support for very high performance graph analytics on large to huge sized graphs. The first component of X-Graphs is SNAP. SNAP provides interactive analytics on graphs with tens of billions of edges that still fit into a single multi-CPU sever memory. The second component of X-Graphs is the Delite framework for building compilers for high-performance Domain Specific Languages (DSLs) that can be used to target heterogeneous architectures (multicore, GPU, cluster, FPGA). These tools are widely used by academia and industry.

15. SUBJECT TERMS
Data Analytics, Graph Analytics, High-Performance Computing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**JOHN SPINA**
U	U	U	SAR	14	19b. TELEPHONE NUMBER (*Include area code*) **(315) 330 4032**

1 SUMMARY

The overall goal of the X-Graphs project was to develop computational techniques and software tools for graph analytics. This report describes the two main components of this project. The first component focuses on support for interactive graph analytics applications on medium to large size graphs. The second component focuses on support for very high performance graph analytics on large to huge sized graphs. The first component of X-Graphs is SNAP [1]. SNAP provides interactive analytics on graphs with tens of billions of edges that still fit into a single multi-CPU sever memory. SNAP 3.0, the most recent release, provides parallel implementations for many key graph algorithms, conversions between tables and graphs and Python language bindings. SNAP is widely deployed with over a thousand downloads per month. The second component of X-Graphs is Delite [18]. Delite is a framework for building compilers for high-performance Domain Specific Languages (DSLs) that can be used to target heterogeneous architectures (multicore, GPU, cluster, FPGA). OptiGraph is a graph DSL that is used to develop graph analytics applications that achieve very high performance on GPUs with small graphs (millions of edges) and also executes on clusters of CPUs with huge graphs (tens of billions of edges). The Delite DSL compiler technology is also capable of targeting the emerging flexible accelerator technology based on FPGAs. Delite is widely deployed and is being used by industry and academia. Delite generated kernels form the core of the DeepDive Knowledge Construction System.

2 INTRODUCTION

The goal of the X-Graphs project was to develop computational techniques and software tools for analyzing massive dynamically changing graphs for new trends, patterns and relationships. Graphs are a powerful way to represent complex data relationships in a compact and efficient fashion. Over the course of the project the goals expanded to encompass the development of software for all components of high-performance data analytics.

Developing data analytics applications composed of massive graphs creates two problems that were solved by the X-Graphs project:

- Problem 1: Software abstractions for developing graph analytic applications. Current graph processing and analysis systems do not work well and are complicated to use due to complicated APIs. Our solution: Graph Domain Specific Languages. The objective here is to expedite the implementation of graph analysis applications via a Graph Domain Specific Language (GDSL). The GDSL will contain the key components of graph analysis algorithms (abstract data structure and algorithmic building blocks) as language elements. During this project, we developed two GDSLs: SNAP.py for fast prototyping of graph algorithms and OptiGraph for high-performance graph analytics.

- Problem 2: High performance platforms for graph processing. Processing graphs in-memory of a single multicore machine calls for parallel graph algorithms.

Processing graphs on distributed shared-nothing architectures requires effective graph partitioning and computation. Our Solution: Heterogeneous architecture. The analyses of large graph streams require large amounts of processing power and no single architecture will be suitable for all problems. Our GDSL compiler will allow us to utilize and optimize graph analytics applications for several different computational architectures and dynamically determine which architecture is most suitable for the different parts of the application. We will consider: (1) semi- streaming platforms where the data arrives as a continuous stream on a single large memory machine, (2) MapReduce/Pregel architecture where the data is stored in a distributed file system.

3 METHODS, ASUMPTIONS, AND PROCEDURES

3.1 Software Abstractions for Graph Analytic Applications

To simplify the development of graph applications we have focused on the development of GDSLs. We have developed SNAP (Stanford Network Analysis Platform), a graph mining and analytics platform, which scales to massive graphs. SNAP is already available as open source (http://snap.stanford.edu). Graph stream algorithms have been integrated with SNAP and made available as part of this platform. We also been developing a GDSL based on the Delite platform for high-performance DSL development. The Delite platform and associated DSLs are available as open source (http://stanford-ppl.github.com/Delite)

Our commercialization plan takes advantage of Stanford's unique connections to Silicon Valley. All the team members already have very strong connections to a variety of local companies, from large to small. We will continuously interact with these companies, as well as with venture capital firms that may fund spin outs. This model has worked very well for the project PIs in the past (generating companies such as Aster Data and Google).

3.2 High performance Platforms for Graph Processing and Data Analytics

The analyses of "big data" requires large amounts of processing power. It is currently a significant burden to develop the best implementations of data analysis algorithms for all the varieties of parallel architecture (multicore, GPU, cluster, FPGA). The traditional library-based approach to this problem has portability and versatility limitations. Instead, we will pursue an approach to developing data-analysis applications based on a suite of domain specific languages (DSLs). We have developed a DSL development framework called Delite to simplify the process of developing high-performance easy to use DSLs. The components of the Delite framework are shown in Figure 1. We have used Delite to develop a suite of DSLs for data analysis (query processing, machine learning, and graph processing).

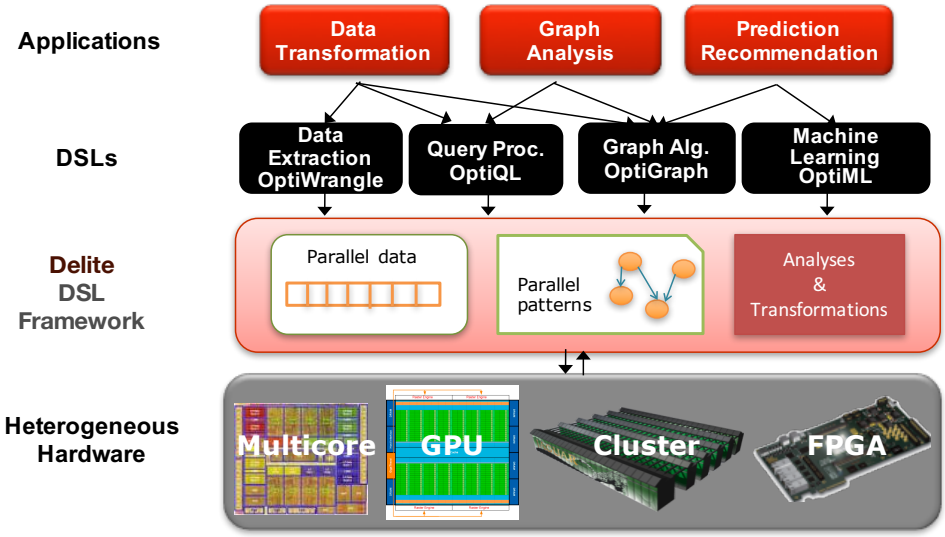

Figure 1. The Delite DSL Framework.

Delite substantially reduces the burden of developing high-performance DSLs by providing common reusable components that enable DSLs to quickly and efficiently target heterogeneous hardware [22].

4 RESULTS AND ACCOMPLISHMENTS

4.1 Software Abstractions for Graph Analytic Applications

Under XDATA, we made our SNAP network analysis software and datasets collection into a robust, open source platform. They became a central resource for network analysts, used by thousands every month.

We developed SNAP into a powerful tool for network analysis. We expanded core SNAP with significant new functionality: built-in operators for relational tables, primitives for graph creation, high-performance implementation of several key algorithms, and support for graphs with attributes. We demonstrated that interactive analysis of networks with billions of edges is practical on modern multi-core large-memory machines. We also provide in SNAP high-performance implementations of novel methods for networks analysis: several methods for detection of overlapping communities, personalized PageRank, node embeddings into a d-dimensional space, counting of temporal motifs, motif-based node clustering. The following new SNAP capabilities have made it accessible to a wide range of users, interested in network analysis: support for Python - a major programming language for data scientists, documentation, tutorials, and unit tests.

The goal of Ringo is Ringo to provide an interactive environment for construction, analysis, and manipulation of graphs on a single large-memory multicore machine. Ringo is based on Snap.py and SNAP, and uses Python. Ringo now allows the integration of

graphs and tables in a single system and provides powerful primitives for graph construction. We received a best demo award at SIGMOD 2015 for the Ringo project.

SNAP network analysis software has been downloaded over 12,000 times in the last 12 months. Its Github repository has 645 stars. The paper on SNAP has 112 citations, according to Google Scholar. The SNAP dataset collection received around 900,000 Web hits in the last 12 months. It has been cited 800 times, according to Google Scholar.

OptiGraph is a DSL, developed under the Delite DSL framework, for static graph analysis based on the Green-Marl DSL [Hong et al. 2012]. OptiGraph enables users to express graph analysis algorithms using graph-specific abstractions and automatically obtain efficient parallel execution. OptiGraph defines types for directed and undirected graphs, nodes, and edges. It allows data to be associated with graph nodes and edges via node and edge property types and provides three types of collections for node and edge storage (namely, Set, Sequence, and Order). Furthermore, OptiGraph defines constructs for Breadth-First Search (BFS) and Depth-First Search (DFS) order graph traversal, sequential and explicitly parallel iteration, and implicitly parallel in-place reductions and group assignments. The SNAP library can be used to generate OptiML programs to take advantage of the high-performance execution options provided by the Delite DSL framework. The OptiGraph publication is cited over 200 times.

4.2 High performance Platforms for Graph Processing and Data Analytics

Under X-Graphs we developed the Delite DSL Framework into a platform for performing large-scale graph analytics, and added new features to improve performance in this domain. The OptiGraph DSL was the main vehicle for our exploration of graph analytics. OptiGraph provides a functional programming model for graph analytics and has demonstrated much better performance with existing state of the art graph processing frameworks such as GraphLab and PowerGraph.

Graph Algorithms – We developed optimized implementations of several important graph algorithms including Betweeness Centrality, Breadth First Search, Page Rank, Triangle Counting, and Community Detection. We spent considerable effort implanting the Louvain Method for community detection for visualizing graphs. These graph algorithms match similar functionality available in SNAP.

Graph Layouts – Besides storing a graph in the well-known compressed sparse row (CSR) representation, we have investigated new ways of laying out a graph in a shared memory environment to enable performance benefits on certain algorithms. In graph algorithms, it is often the case that a join query of some nature is required to provide rapid lookup of a node in a neighborhood. We have developed a platform for OptiGraph that parametrically allows graph neighborhoods as well as computation frontiers to be stored and computed in several different graph data layouts. We have devised a methodology to decide which algorithmic and dataset dependent factors make one data-layout better than another. Armed with this methodology, we developed algorithms in the

OptiGraph compiler that automatically choose the optimal data-layout.

Graph Scheduling – Delite has historically allowed for only statically scheduled parallel computations, but due to the load imbalance performance losses that can occur with graph computations we extended the functionality of our scheduler. First, we are implementing a dynamic scheduler that will effectively provide work stealing across any parallel loops in Delite. Extending this idea further, we have added an asynchronous engine in Delite that allows graph computations to extend across iterations of parallel computations without any need for costly synchronizations. Several existing graph processing platforms have shown that doing this provides a performance advantage for certain graph algorithms.

We made many enhancements to the general Delite infrastructure, both in terms of making Delite DSLs easier to create and in terms of more powerful optimizations and improved performance. The Delite compiler initially only had fixed phases, and DSL authors could only perform optimizations as the IR was being constructed. We have now added support for DSL authors to define additional IR traversals and transformations. Using this support, we implemented powerful data structure optimizations that were not feasible without this facility, including array-of-struct to struct-of-array transformations and dead field elimination. These optimizations are made possible by introducing a Record type as a first-class citizen in Delite that both DSL authors and DSL users can extend to make custom records/structs. These optimizations combined with an improved parallel loop fusion algorithm and some other existing optimizations have produced order of magnitude speedups across multiple applications [1].

We developed support in Delite for mapping to non-CPU accelerators such as GPUs. A key element of this expansion is our C++ code generation coverage of entire applications. This support has many practical benefits including the ability to generate vectorized code, generate NUMA-aware code within a single process, allocate very large arrays, target new accelerators such as the Xeon Phi, and communicate between the CPUs and accelerators with less overhead (i.e., copying memory in and out of the JVM). To make this feasible we created our own custom garbage collection mechanism for Delite applications using a combination of static analysis and dynamic reference counting. Our high-level analysis in Delite provides all the information required to generate vectorizable loops. We can communicate this information to the C compilers using the OpenMP *SIMD* pragma and let the compiler do the detailed work of code generation for X86 or Xeon Phi vector units. Support for the *SIMD* pragma will soon be available for the GCC and LLVM compilers with the release of OpenMP 4.0.

We have complete support in Delite for multi-dimensional mapping to hierarchical parallelism in GPUs. This support provides a 28.6x speedup over 1D mappings and 9.6x speedup over existing 2D mappings. This support makes Delite generated code equal to or better than hand optimized CUDA code.

Delite is now built around a high-level parallel intermediate language called DMLL the Delite Multiloop Language (DMLL). DMLL can be used to generate efficient code for

various combinations of devices (including C++, CUDA, and OpenCL). DMLL provides implicitly parallel collection operations but lacks "difficult" features such as higher order functions, recursion, or an object system. This enables high-level analyses and transformations, such as sophisticated loop fusion and data access optimizations.

The DMLL has been used to support two new analysis and optimization techniques that can dramatically improve performance of DSL programs.

- Partitioning Analysis which decides which data structures and parallel operations to partition across multiple memory regions in a distributed memory or NUMA architecture.
- Nested pattern transformations that optimize patterns for distributed heterogeneous architectures to ensure data is consumed and produced

Using DMLL we now support in Delite for NUMA architectures. This support makes it possible to run machine leaning algorithms 10-50 times faster on a 48 core NUMA architecture compared to well-known analytic environments like SPARK and PowerGraph. Using DMLL we improved the OptiGraph implementation of Gibbs sampling so that it is now 4x better than DimmWitted. Gibbs sampling is the main inference algorithm used in DeepDive. We integrated the OptiGraph Gibbs sampler into DeepDive. This involves replacing the hand coded C++ Gibbs sampler (DimmWitted) in DeepDive with the one generated from OptiGraph DSL code. The result of this accomplishment is simpler code and much higher performance (4x) than the hand coded C++ version.

Towards the end of the X-Graphs project we explored using Delite to automatically generate hardware for DSL applications by adding the compiler analyses and transformations necessary to create implementations that are far more sophisticated than traditional C-to-Verilog systems. To more precisely reason about data structure usage, we added first-class multidimensional arrays to Delite. With this abstraction, we can analyze data access patterns in a much more straightforward way than mapping everything to flat arrays. With this new information, we implemented automatic tiling transformations on nested parallel patterns to greatly improve locality by storing reused data in local memories. While this transformation is applicable to many hardware architectures, we found it particularly essential for targeting FPGAs which lack built-in caches and memory pre-fetchers. We then added code generators from Delite parallel patterns to HDL modules. In addition to simply mapping each tiled Delite parallel pattern to a hardware design, we also automatically inserted fine-grained hierarchical pipelining (metapipelining) across Delite operations and at each nesting level to greatly improve design throughput.

We developed a design framework using this new Delite representation of hardware using parameterized templates that captures locality and parallelism information at multiple levels of nesting. This representation, called Spatial, is designed to be automatically generated from high-level intermediate languages based on parallel patterns such as DMLL. We have developed a hybrid area estimation technique which uses template-level models and design-level artificial neural networks to account for effects from hardware place-and-route tools, including routing overheads, register and

block RAM duplication, and LUT packing. Our runtime estimation accounts for off-chip memory accesses. We use our estimation capabilities to rapidly explore a large space of designs across tile sizes, parallelization factors, and optional coarse-grained pipelining, all at multiple loop levels. We show that estimates average 4.8% error for logic resources, 6.1% error for runtimes, and are 279 to 6533 times faster than a commercial high-level synthesis tool. We compare the best-performing designs to optimized CPU code running on a server-grade 6 core processor and show speedups of up to 16.7×.

We have developed the Spatial IR into a new full-fledged performance oriented programming language called *Spatial*. Spatial is intended for performance engineers who care about optimizing the performance of applications by specifying parallelism and locality. The language is especially valuable for the emerging class of configurable accelerator architectures such as field programmable gate-arrays (FPGAs) and coarse-grain reconfigurable architecture (CGRAs) that are being used to accelerate applications in data analytics and machine learning. The Spatial compiler uses techniques developed in the Delite compiler to translate DSL applications to hardware using the Chisel HDL. We have demonstrated much 2–5 times the performance improvement compared to existing FPGA compilers that convert C to hardware using high-level synthesis. The growing use of FPGA based accelerators for machine-learning and data processing by companies such as Microsoft and Amazon suggest that the Spatial compiler technology will play a key role in the data-analytics accelerator market.

The Delite DSL Framework and associated DSLs (OptiGraph, OptiML, and OptiQL) are available as open source on GitHub. The Delite technology is broadly used by academia and industry. Industrial use includes Oracle, Intel and a few start-up companies.

5 CONCLUSIONS

The overall goal of the X-Graphs project was to develop computational techniques and software tools for graph analytics. We were successful in achieving this goal with the development of two key software technologies. SNAP a system designed for interactive analytics on large graphs that fit into the memory of a single server and Delite a DSL compiler framework for optimizing DSLs to heterogeneous computer architectures. Taken together these technologies allow a Data Scientist developer to interactively develop an analytics application using SNAP and then deploy it using Delite with the goal of achieving much high-performance with much larger dataset sizes. Both SNAP and Delite are widely-used open source software available on GitHub. The X-Graphs project has supported the generation of over thirty peer-reviewed technical publications.

6 PUBLICATIONS SUPPORTED BY X-GRAPHS

1. J. Leskovec and R. Sosič. “Snap: A general-purpose network analysis and graph-mining library.” *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 8, no. 1, July 2016.

2. C. R. Aberger, S. Tu, K. Olukotun, and C. Ré, "EmptyHeaded: A Relational Engine for Graph Processing," *SIGMOD '16: Special Interest Group on Management of Data*, June 2016. (Best of Award)

3. C. De Sa, K. Olukotun, and C. Ré, "Ensuring Rapid Mixing and Low Bias for Asynchronous Gibbs Sampling," *ICML '16: Proceedings of the 33rd Intl. Conference on Machine Learning*, June 2016. (Best Paper Award)

4. D. Koeplinger, R. Prabhakar, Y. Zhang, C. Delimitrou, C. Kozyrakis, and K. Olukotun "Automatic Generation of Efficient Accelerators for Reconfigurable Hardware," *ISCA '16: 43rd International Symposium on Computer Architecture*, June 2016.

5. R. Prabhakar, D. Koeplinger, K. J. Brown, H. Lee, C. De Sa, C. Kozyrakis, and K. Olukotun, "Generating Configurable Hardware from Parallel Patterns," *ASPLOS '16: 21st International Conference on Architectural Support for Programming Languages and Operating Systems*, April 2016.

6. K. J. Brown, H. Lee, Tiark Rompf, A. K. Sujeeth, C. De Sa, C. Aberger, and K. Olukotun, "Have Abstraction and Eat Performance, Too: Optimized Heterogeneous Computing with Parallel Patterns," *CGO '16: International Symposium on Code Generation and Optimization*, March 2016.

7. T. Oguntebi and K. Olukotun, "GraphOps: A Dataflow Library for Graph Analytics Acceleration," *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, February 21-23, 2016.

8. C. De Sa, C. Zhang, C. Ré, and K. Olukotun, "Taming the Wild: A Unified Analysis of Hogwild!-Style Algorithms," *NIPS '15: Proceedings of the 28th Neural Information Processing Systems Conference*, December 2015.

9. C. De Sa, C. Zhang, C. Ré, and K. Olukotun, "Rapidly Mixing Gibbs Sampling for a Class of Factor Graphs Using Hierarchy Width," *NIPS '15: Proceedings of the 28th Neural Information Processing Systems Conference*, December 2015.

10. N. George, H. Lee, D. Novo, M. Owaida, D. Andrews, K. Olukotun and P. Ienne, "Automatic support for multi-module parallelism from computational patterns," *FPL '15: Proceedings of Field Programmable Logic 2015*, London, UK, August 31-Sept 4, 2015.

11. C. De Sa, K. Olukotun, and C. Ré, "Global Convergence of Stochastic Gradient Descent for Some Non-convex Matrix Problems," *ICML '15: Proceedings of the 32nd Intl. Conference on Machine Learning*, July 2015.

12. R. Sosic, A. Banerjee and, R. Puttagunta, M, Raison, P. Shah and J. Leskovec, "Ringo: Interactive Graph Analytics on Big-Memory Machines," *Proceedings of*

the 2015 ACM International Conference on Management of Data (SIGMOD), June 2015.

13. T. Rompf, K. J. Brown, H. Lee, A. K. Sujeeth, K. Olukotun, "Go Meta! A Case for Generative Programming and DSLs in Performance Critical Systems," *SNAPL 2015: Symposium on Advances in Programming Languages,* May 3–4, 2015.

14. L. McAfee and K. Olukotun, "EMEURO: A Framework for Generating Multi-Purpose Accelerators via Deep Learning," *CGO '15: International Symposium on Code Generation and Optimization*, San Francisco, CA, Feb. 10, 2015.

15. K. Olukotun and L. Hammond, "Author's retrospective for: Improving the performance of speculatively parallel applications on the Hydra CMP," *ACM International Conference on Supercomputing 25th Anniversary Volume,* 2014.

16. H. Lee, K. J. Brown, A. K. Sujeeth, T. Rompf and K. Olukotun, "Locality-Aware Mapping of Nested Parallel Patterns on GPUs," 47th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2014, Cambridge, United Kingdom, December 13-17, 2014.

17. N. George, H. Lee, D. Novo, T. Rompf, K. J. Brown, A. K. Sujeeth, M. Odersky, K. Olukotun and P. Ienne, "Hardware system synthesis from Domain-Specific Languages," *4th International Conference on Field Programmable Logic and Applications, FPL 2014*, Munich, Germany, 2-4 September 2014.

18. K. Sujeeth, K. J. Brown, HyoukJoong Lee, Tiark Rompf, Hassan Chafi, Martin Odersky, and Kunle Olukotun, "Delite: A Compiler Architecture for Performance-Oriented Embedded Domain-Specific Languages," *TECS'14: ACM Transactions on Embedded Computing Systems*, July 2014.

19. T. Rompf, A. K. Sujeeth, K. J. Brown, H. Lee, H. Chafi and K. Olukotun, "Surgical precision JIT compilers," ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, June 09 - 11, 2014.

20. J. Casper and K. Olukotun, "Hardware acceleration of database operations, "2014 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '14, Monterey, CA, USA, February 26 - 28, 2014

21. S. Hong, S. Salihoglu, J. Widom and K. Olukotun, "Simplifying Scalable Graph Processing with a Domain-Specific Language," *12th Annual IEEE/ACM International Symposium on Code Generation and Optimization, CGO '14*, Orlando, FL, USA, February 15-19, 2014.

22. J. Yang, J. McAuley, and J. Leskovec. “Community detection in networks with node attributes.” *2013 IEEE 13th international conference on Data Mining (ICDM),* December, 2013.

23. S. Hong, N. C. Rodia, K. Olukotun, “On fast parallel detection of strongly connected components (SCC) in small-world graphs,” Supercomputing, November 2013.

24. K. Sujeeth, A. Gibbons, K. J. Brown, H. Lee, T. Rompf, M. Odersky, K. Olukotun, “Forge: generating a high-performance DSL implementation from a declarative specification,” *GPCE'13: 12th International Conference on Generative Programming: Concepts & Experiences*, October 2013.

25. K. Sujeeth, T. Rompf, K. J. Brown, H. Lee, Hassan Chafi, V. Popic, M. Wu, A. Prokopec, V. Jovanovic, M. Odersky, and K. Olukotun, “Composition and reuse with compiled domain-specific languages,” *ECOOP'13: European Conference on Object-Oriented Programming*, July 2013.

26. T. Rompf, A. K. Sujeeth, N. Amin, K. J. Brown, V. Jovanovic, H. Lee, M. Jonnalagedda, K. Olukotun, M. Odersky, “Optimizing data structures in high-level programs: new directions for extensible compilers based on staging,”*POPL'13: 40th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, January 2013.

27. S. Hong, T. Oguntebi, J. Casper, N. Bronson, C. Kozyrakis and K. Olukotun, “A Case of System-level Hardware/Software Co-design and Co-verification of a Commodity Multi-Processor System with Custom Hardware,” *CODES+ISSS'12: 17th International Conference on Hardware/Software Codesign and System Synthesis*, Oct 2012.

28. L. McAfee, K. Olukotun, “Utilizing Static Analysis and Code Generation to Accelerate Neural Networks,” *International Conference on Machine Learning (ICML).* June 2012.

29. H. Chafi, A. K. Sujeeth, Z. DeVito, K. Olukotun, “High-Performance Domain-Specific Languages with Delite,” *International Parallel & Distributed Processing Symposium (IPDPS)*, Shanghai, China, May 2012.

30. S. Hong, H. Chafi, E. Sedlar, and K. Olukotun, “Green-Marl: A DSL for Easy and Efficient Graph Analysis,” *17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012)*, March 2012.